PILOT II

PILOT II has many new features. These are described below under six general categories.

TEXT AND INPUT - including large, medium, and small letters, single character input, and sensing of keyboard use.

SOUND - including relative notation and combined sound and duration.

GRAPHICS - including access to all graphics modes, full or split screen, full control of color for pen and background, screen maze walls, turtle behavior at walls and screen edges, and general graphics changes.

DEBUGGING TOOLS - including tracing, single-stepping, execution break and continue, and dumps and clearing of string and numeric variables.

PROGRAM MANAGEMENT - including loading, merging, appending, renumbering, and deleting programs or program segments, formatted listings, auto line numbering, and getting disk directories.

OTHER FEATURES - including execution speed, saving and restoring screens, numeric variable names, and robot turtle controls.

1. Large letter text screen option -- the LETTERS command with operand options of SMALL, MEDIUM or LARGE, will invoke screen modes 0, 1 or 2 for the text mode screen.

```
example:  LETTERS:LARGE
          T:HERE ARE SOME LARGE LETTERS
```

The POSition, Type and Accept commands will have the following new behaviors when the letter size is selected to be MEDIUM or LARGE:

The Accept command will error; however, the AK command may be used to get input directly from the keyboard (rather than from the screen editor) and the keystrokes will not be displayed.

The screen will not scroll, the POS command must be used to reposition the text cursor.

Screen editing output function characters (tab, cursor up, etc.) will display rather than being acted upon.

Large letters are to be implemented primarily as an output feature.

The SETLET statement allows you to set the color of medium or large letters. The form of this statement is:

```
   SETLET:  <color register>  ,  <hue>  ,  <lum>
or SETLET:  <color register>  ,  <color name>

 <color register>  ::= 0      [background
                       1 - 4  [four colors
```

```
example:  SETLET: 1, 6, 8 puts a hue of 6 and luminance of 8 in
          color register 1 without going into graphics mode (see
          SETPEN).
```

To get the four different colors, use different types of characters in the T: statement. Upper case letters give you color 1, lower case is color 2, inverse video upper case is color 3, and inverse video lower case is color 4.

The graphics mode text window will always be small letters.

Any text output by LIST, TRACE, DUMP, etc. will force the screen to the standard small letter text screen.

2. POS will honor the row number for the text window in graphics mode.

3. Keyboard character ready sense -- %K will return the value 0 (false) if there is no keyboard character to be read, and will return the value 1 (true) if there is a keyboard character to be read.

```
example: J(%K):*INTERRUPT
```

4. Read K: one character at a time -- the new command AK, meaning Accept Key, reads a single keystroke into the accept buffer. Upon command completion the accept buffer will contain a space, the ATASCII code for the key pressed, followed by a space.

```
example:AK:                [read key and see if Y or N.
        M:Y,N
```

5. Accept literal -- the new command AX will read data to the Accept buffer with none of the conversion rules applied.

1. "No change" symbol -- when a note in the note list is not to change from its current value, the symbol '=' can be used.

```
example: S0:1,5,8,13          [ C major chord
         S0:=,4,=,=           [ same as S0: 1,4,8,13
```

2. Self-relative (turtle music) notation -- when a note in the note list is to change by a specific interval from its current value, the change may be specified in relative notation using the '+' and '-' symbols, as shown below:

```
example: S0:1,5,8,13          [ C major chord
         S0:=,-1,=,=          [ C minor chord, flat the 3rd
         S0:=,+1,=,=          [ C major chord again
```

3. Integrate the note duration with the sound command -- augment the S0 command syntax as shown:

```
< command >  ::=S0:  < operands >
< operands > ::=  < note list >    (  < note list >  )  < duration >

< note list > ::= [ < note >  [ < note >  [ < note >  [ < note >  ] ] ] ]
< note >   ::=  < nvar >  [  < const >  [  < pointer >
< duration > ::=  < nexp >
examples: S0:1,5,8,13        [ play notes and go to next stmt
          S0:(1,5,12) 30     [ play notes and pause 30 ticks
```

4. S0 operand evaluation -- S0 command operands are now always evaluated to produce a note value, never the address of a note value. As a consequence, the construct shown below has a different effect in new PILOT than in 8K PILOT.

```
C:#A=1
S0:#A
PA:60
C:#A=5              [ changes tone in 8K PILOT; not in new PILOT
PA:60
```

1. Implied GR: graphics subcommands may be entered without the GR: in all contexts. Leading iterations must be preceded by the subcommand name REPEAT so as not to be confused with the entry of a numbered line.

2. New visible turtle with 15° resolution.

3. Visible turtle control -- the TURTLE graphics subcommand will control the presence of the visible turtle with the operands ON and OFF. The same command can be used to change the color of the turtle.

   example: GR: TURTLE ON; GO 10
   example: GR:TURTLE RED

4. Home and North -- the graphics subcommands HOME and NORTH will have the effect of GOTO 0 0 and TURNTO 0, respectively.

5. LOGO synonyms -- Several of the PILOT graphics subcommands have LOGO synonyms, as shown below.

PILOT subcommand	LOGO synonym
DRAW n	FD n
TURN n	RT n
n	REPEAT n
DRAWTO x,y	SETPOS x,y
TURNTO n	SETH n
CLEAR	CLEAN
BACKGROUND c	SETBG c
DRAW -n	BK n
TURN -n	LT n
PEN UP	PU
PEN DOWN	PD
PEN ERASE	PE

6. Graphics mods selection -- The MODE subcommand allows the selection of any of the standard Display Handler screen modes. The table below shows the attributes of the various modes.

PILOT MODE	ANTIC MODE	COLS	ROWS	FULL/ SPLT	PENS
3	8	40	24	both	3
4	9	80	48	both	1
5	A	80	48	both	3
6	B	160	96	both	1
7	D	160	96	both	3
8	F	320	192	both	1 (lum only)
9	F+GTIA	80	192	full	15 lums, 1 hue
10	F+GTIA	80	192	full	8
11	F+GTIA	80	192	full	15 hues, 1 lum
14	C	160	160	both	1
15	E	160	160	both	3

Note that modes 9-11 require the selection of FULL screen prior to the mode selection.

Modes 14 and 15 are not available on the ATARI 400 and 800.

The graphics screen is cleared whenever there is a change in the mode number, otherwise it is not cleared. All executions of the MODE subcommand cause the turtle to go to the home position and to be visible.

Because of the special color capabilities of graphics modes 8, 9 and 11, they must be treated differently than the other modes, as shown below.

```
MODE 8 -- SETPEN:0,  < hue >  , < lum >
[ sets hue and lum for background
          SETPEN:1,0,  < lum >          [ sets lum for the single pen
          PEN ERASE or PEN 0            [ sets pen erase
          PEN 1                         [ selects the single pen

MODE 9 -- SETPEN:0, < hue >,0           [ sets single hue for mode
          PEN n                         [ n = 0-15, selects lum 0-15

MODE 11-- SETPEN:0,0, < lum >           [ sets single lum for mode
          PEN n                         [ n = 0-15, selects hue 0-15
```

7. Make text window optional -- the graphics subcommands FULL (for full graphics screen) and SPLIT (for split screen) allow the user to go back and forth between the two screen modes without clearing the display memory. If not specified otherwise, the graphics screen comes up split. Any access to the screen editor (Type, Accept, etc.) while in a full graphics screen will force the graphics screen to split.

```
example:  GR:FULL; CLEAR; PEN RED
```

8. Color selection -- the graphics subcommand PEN will be augmented to allow general color selection from the palette of colors than can be generated from the GTIA chip. The turtle will have a variable number of pens, depending upon the graphics mode selected; graphics mode 7, for example, will have 3 pens for foreground colors, plus a background color. The system will be initialized with no colors assigned to the pens, and as the user requests a color, an unassigned pen will be used until all of the pens are assigned a color. Once all of the pens are assigned a color, the user may explicitly CHANGE one of the existing pen colors to a new color whenever he wants to utilize a new color, or he may clear the pen color assignments by using the CLEARPENS subcommand.

   A new graphics subcommand BACKGROUND will allow the user to assign any color to the background.

   example:

```
GR:PEN BLUE;  DRAW 5     [ select pen and set to BLUE
GR:PEN GREEN; DRAW 5     [ select pen and set to GREEN
GR:PEN RED;   DRAW 5     [ select pen and set to RED
GR:PEN BLUE;  DRAW 5     [ reselect the BLUE pen
GR:CHANGE BLUE,ORANGE; DRAW 5  [ change BLUE to ORANGE
GR:BACKGROUND YELLOW     [ set the background to YELLOW
GR:CLEARPENS             [ clear all pen color assignments
```

9. Full color control -- the new command SETPEN (similar to BASIC'S SETCOLOR, but not identical) allows selection of colors in numeric form (by hue and luminosity).

   < operand > ::= < pen number >, < hue > , < intensity >

   where: < pen number > ::= 0-n (0 is always the background, n is a function of the mode #)
   < hue > ::= 0-15
   < intensity > ::= 0-7

   example: SETPEN:1,14,5

10. Pen color variables -- a numeric expression may be used for pen color selection; the value is a pen number. The number of pens available is a function of the screen mode. Pen number zero is the background.

   examples: GR:PEN #A GR:PEN 4-*C GR:CHANGE 2,BLUE

11. Region shading -- the graphics subcommand SHADE will change the color of the pixel under the turtle to the indicated color, and will change the color of all adjoining pixels of the same color. The pen may be up or down for this to work. SHADE has a single operand, which is the shading color selection.

    example: GR:PEN UP; GO 5; SHADE BLUE

12. COLORS command -- The COLORS command will list the names of the built in colors to the screen. These names may be used as operands in the PEN, BACKGROUND, SHADE, CHANGE AND WALL subcommands, and in the SETPEN command.

13. Pen Down -- the PEN operand DOWN will lower the turtle pen using the last selected color.

    PEN color selection implies DOWN.

14. %N returns the pen status (pen number, + 128 if up).

15. PS command -- The PS (Pen Status) command will list the status of the turtle pens to the graphics text window, in the form shown below.

```
PENS: 1=RED  2=ORANGE  3=
BACKGROUND:BLACK            MODE: 7
```

16. Graphics mode initialization -- When the screen changes from a text mode to a graphics mode, the following turtle environment is established.

```
MODE 7
SPLIT
CLEAR
CLEARPENS
BACKGROUND BLACK
PEN ERASE
PEN DOWN
HOME
NORTH
TURTLE ON
EDGE FREE
WALL NONE
```

17. Walls -- the graphics subcommand WALL will allow the user to select up to three of the pen colors to be treated as walls. A current pen color or the pen numbers 1-3 will establish that value as the current wall color. The operand NONE will disable the wall logic.

    example: GR:WALL RED

18. Screen edge options -- the graphics subcommand EDGE has the following allowable operands:

    WRAP -- Turtle wraps screen edges and stops at walls.
    HALT -- Turtle stops at screen edges and walls.
    BOUNCE -- Turtle bounces (reflects) off screen edges and stops at walls.
    FREE -- Turtle ignores screen edges and walls (a la 8K Atari PILOT).

    example: GR:EDGE BOUNCE; TURNTO 43; DRAW 30000

19. Turtle sensor -- %ST will return the graphics wall sensing status; the graphics turtle has a single "nose sensor" which will return a vlaue of

    2 - turtle at screen edge or beyond

    1 - turtle at wall

    0 - neither of the above

20. Speed control of outputs -- the DELAY command will meter the rate at which screen outputs are transmitted and will control the speed of the turtle.

    example: DELAY:3

21. ES command reports on edge rule, speed and wall status.

1. Stop program -- the STOP command will cause an executing program to stop in a continuable form.

2. Continue program -- the CONT command will allow the user to continue execution after a STOP command, an operator BREAK and after some run-time errors. The user may not continue after any program editing has occurred.

3. Single step of program -- pressing the START key will cause a single statement to be executed and displayed to the text screen. If the program is continuable, the next statement will be executed; if the program is not continuable, execution will start at the lowest numbered line. If a program is executing, the START key will cause the same action as the BREAK key.

4. Statement trace control -- pressing the OPTION key will turn the statement trace alternately on and off, while a program is executing.

5. DUMP command change -- The DUMP command now accepts an optional operand of 'S' or '#' to select the listing of only the string variables or the numeric variables, respectively.

6. VNEW command change -- The VNEW command no longer has the optional operand to specify selective clearing of the string variables or numeric variables. VNEW now clears both sets at all times.

1. Disk directory list -- The DIR command lists the directory of the specified disk to the screen.

   example: DIR 1 [lists the directory of drive 1.

2. LOAD, APPEND, MERGE -- The LOAD command now clears the program storage area in immediate mode as well as run mode. The new commands APPEND and MERGE may be used to merge program segments.

   MERGE < filename > merges the indicated file with the resident program, replacing any identically numbered statements with the statements from the file.

   APPEND < filename > loads the statements from the indicated file, while renumbering them to reside at the end of the program storage area.

3. Renumber command change -- The REN command now has two additional operands to allow it to move code segments. The new operand list syntax is shown below.

   < operand list > ::= [< target line # > [, < target # incr >
   [, < source start # > [, < source end # >]]]]

   A few examples are shown below.

```
REN                 [ renumbers by 10s starting at 10.
REN 100             [ renumbers by 10s starting at 100
REN 30,20           [ renumber by 20s starting at 20.
REN 500,10,300      [ renumbers the statements between 300 and
                      the end of the program to 500, by 10s.
REN 900,10,60,90    [ renumbers the statements between 60 & 90
                      to 900, by 10s.
```

   Renumber will not renumber any lines unless they all may be correctly renumbered.

4. DEL command - The DEL command allows the user to delete any number of stored program lines. The syntax is shown below.

   < operand list > ::= < start line # > [, < end line # >]

   A few examples are shown below

```
DEL 50              [ delete line 50.
DEL 80,210          [ delete lines 80 through 210, inclusive.
```

5. AUTO default line number -- The default starting line number for AUTO is now ten past the highest numbered line in the program storage area, or 10 if no line is present.

6. LIST format -- The LIST command now displays the line number as a five digit field, with the line number right justified with leading blanks. This is to aid the user in forming consistent indentation that is independent of the number of digits in the line number.

7. Leading spaces are retained on statement storage.

PILOT II - OTHER FEATURES

1. Save/restore screen -- the SSAVE and SLOAD commands save and restore graphics or text screen data and mode to a file or from a file to the screen. These commands are available for run or immediate mode execution.

2. General numeric variable names -- allow the same naming convention for numeric variables as is used with string variables (substituting '#' for '$'). All letters in the name are significant.

   examples: #COUNT #ANGLE

3. Robot turtle hooks -- The robot turtle will be driven through a standard single entry interface. The interface commands will include:

   Robot go forward/backward n units.
   Robot turn left/right n degrees.
   Robot return sensor data.

4. Lower case/upper case equivalence -- in all PILOT defined symbols (commands, subcommands, conditionals, special variables, literal operands, etc.) lower case letters will be treated as upper case letters when doing name matching. In all user defined symbols (variables and labels), the upper case letter set and the lower case letter set are distinct.

   example: ty:Free memory is %f bytes.
   r:#COUNT is not the same as #count.

5. %(< nexp >) -- forces an evaluation of the < nexp > within any variable allowed context.

   example: T:#VAL1 + #VAL2 = %(#VAL1+#VAL2).

6. %" < text literal > " -- allows the text between the quotes to be treated literally, rather than be evaluated, within a text expression.

   example: T:THE VALUE OF %"#A" IS #A.

7. Better controller variable syntax -- allow numeric expressions in the controller specification.

   < controller > ::= % < alpha > < nexp >

   examples: %J3 %T#A %P(#A+1)

8. Use stack depth increase -- The Use stack depth has been increased from 8 to 24 levels.

9. SCROLL command -- The SCROLL command accepts operands of COARSE or FINE and selects the corresponding scrolling mode using the Screen Editor. This command has no effect on the ATARI 400 and 800.

10. Lightpen removal -- All lightpen support has been removed.

11. Tokenized commands -- all command names will be tokenized during the statement input phase (edit, ATUO, LOAD, APPEND or MERGE).

12. TV command -- you can turn the video display on or off with this command. Use TV:ON or TV:OFF.